REMARKS

Applicant respectfully requests reconsideration of the rejected claims. Claims 1, 4-8, 11-13, 15-21, 23-30, 32, 35-39, 42-44 remain in the application. Claims 2, 3, 9, 10, 14, 22, 31, 33, 34, 40, and 41 have been canceled. No claims have been added.

Claims 1, 8, 13, 21, 28, 32, and 39 have been amended to better clarify the scope of the embodiments claimed, thus Applicant added no new matter.

Applicant amended paragraph [0039] in the specification affect the readability of a sentence not its content. Thus, the amendment added no new matter.

Claim Rejections under 35 U.S.C. § 102(b)

Claims 1, 4-5, 7, 13, 17-19, 21, 23, 28, 30, 32, 35-36, and 38 have been rejected under 35 U.S.C. §102(b) as being anticipated by Schneier et al. "NPL Fast Software Encryption: Designing Encryption Algorithms for Optimal Software Speed on the Intel Pentium Processor" ("Schneier").

Applicant respectfully disagrees with the rejection because Schneier does not describe each and every element of the rejected claims. Applicant requests reconsideration of the rejected claims.

Claim 1

Claim 1 requires a method including receiving a data cipher operation and

processing the data cipher operation. The processing includes "generating a number of portions of ciphertext from plaintext." The processing also includes "a load operation associated with the generating of at least **one portion** of the ciphertext" that "executes **prior** to a store operation associated with the generating of a **prior portion** of the ciphertext." Furthermore, "the generating of the at least one portion of the ciphertext and the generating of the prior portion of the ciphertext is executed **within one iteration** of a number of iterations for the data cipher operation, and wherein the generating of the at least one portion of the ciphertext is **re-executed** in an iteration that is subsequent to the one iteration upon determining that data retrieved from the load operation conflicts with data stored in the store operation."

Schneier describes the inner loop of an RC4 stream cipher that XORs 8 bits of keystream data. This prior art algorithm, as described in Schneier and disclosed as the prior art in Applicants' application in paragraphs [0006]-[0024], loads a values S[i] and S[j] from the S-box array and assigns the values to variables tmpI and tmpJ respectively. Next, the prior art algorithm stores the values of the variables tmpI and tmpJ into the S-box array in locations S[j] and S[i] respectively, resulting in the swapping of the values of S[i] and S[j]. Following this, the prior art algorithm adds the values of tmpI and tmpJ and assigns this value to a variable t. Next, the prior art algorithm employs the value t as an index to reference an element in the S-box array. The prior art algorithm uses this t element of S-box array to XOR the 8 bit keystream into ciphered data.

Hence, the inner loop of this prior art algorithm only loads the values from the

S-box needed to calculate the value of the cipher for one 8 bit keystream. In other words, the stores associated with one portion of 8 bit keystream are completed before a load associated with another portion of 8 bit keystream.

Schneier further describes the technique of unrolling this prior art algorithm to prevent a pipeline stall because of the address set up time required for table indexing. **Schneier describes that in a fully unrolled optimized loop of this prior art algorithm "the optimizations include carefully overlapping the start of the next iteration with the end of the previous one."** (Schneier, pg. 248). This careful overlapping is necessary because as Schneier describes, "almost every statement depends immediately on the statement before it, including the table index computation and the associated table accesses, limiting the amount of parallelism achievable." (Schneier, pg. 248). The unrolling of the loop must be properly overlapped because **"the overlapping of the generation of the two different portions of cipher data are non-speculative in nature"**; therefore, "to avoid the generation of inaccurate data for the ciphertext, the write operation to the S-box for the generation of a first portion of ciphertext is complete prior to the load operation to the S-box for the generation of a second portion of ciphertext." (Application, para. [0024]). Thus, even in a fully unrolled optimized loop of the algorithm as described by Schneier the stores associated with an 8 bit portion of keystream must be completed before a load associated with another 8 bit portion of keystream to prevent the algorithm outputting unusable ciphertext.

Furthermore, Schneier fails to describe the generation of the **at least one portion** of the ciphertext and the generating of **the prior portion** of the ciphertext is

executed **within one iteration** because the inner loop of this prior art algorithm only loads the values from the S-box needed to calculate the value of the cipher for one 8 bit keystream. Moreover, Schneier fails to describe one portion of the ciphertext is **re-executed** in an iteration that is subsequent to the one iteration upon determining that data retrieved from the load operation conflicts with data stored in the store operation. In other words, Schneier fails to describe generating two portions of ciphertext within one iteration and re-execution of one portion under certain conditions.

Therefore, Schneier fails to describe processing a data ciphering operation as required in claim 1 "wherein the processing comprises generating a number of portions of ciphertext from plaintext, wherein a load operation associated with the generating of at least **one portion** of the ciphertext executes **prior** to a store operation associated with the generating of a **prior portion** of the ciphertext." Furthermore, Schneier fails to describe that "one portion of the ciphertext is re-executed in an iteration that is subsequent to the one iteration upon determining that data retrieved from the load operation conflicts with data stored in the store operation."

These aspects of claim 1 increase the number of process cycles that the memory is accessed over the prior art algorithm as disclosed in Schneier. Applicant's claim 1 more fully utilizes the memory and provides for faster execution of data ciphering over that disclosed in Schneier. These aspects can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Because Schneier does not describe a data ciphering operation as in claim 1 "wherein the processing comprises generating a number of portions of ciphertext from plaintext, wherein a load operation associated with the generating of at least **one portion** of the ciphertext executes **prior** to a store operation associated with the generating of a **prior portion** of the ciphertext, . . . the generating of the at least one portion of the ciphertext and the generating of the prior portion of the ciphertext is executed **within one iteration** of a number of iterations for the data cipher operation, and wherein the generating of the at least one portion of the ciphertext is **re-executed** in an iteration that is subsequent to the one iteration upon determining that data retrieved from the load operation conflicts with data stored in the store operation," Schneier fails to anticipate claim 1.

Furthermore, Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claim 1 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claims 4-7

Applicants respectfully submit that claims 2-7 are dependent directly or indirectly on claim 1, thus include the same limitations as claim 1. As such, claims 2-7 are patentable for at least the same reasons as claim 1.

Claim 32

Applicant respectfully submits that claim 32 as amended requires similar

limitations as claim 1. Specifically, claim 32 as amended requires a machine-readable medium that provides instructions including receiving a data cipher operation and processing the data cipher operation. The processing includes "generating a number of portions of ciphertext from plaintext" such that "a load operation associated with the generating of at least **one portion** of the ciphertext executes **prior** to a store operation associated with the generating of a **prior portion** of the ciphertext." Furthermore, claim 32 requires "the generating of the at least one portion of the ciphertext and the generating of the prior portion of the ciphertext is executed **within one iteration** of a number of iterations for the data cipher operation and wherein the generating of the at least one portion of the ciphertext is **re-executed** in a iteration that is subsequent to the one iteration upon determining that data retrieved from the load operation conflicts with data stored in the store operation."

Thus, for at least the reasons discussed for claim 1, Schneier fails to anticipate claim 32.

Furthermore, Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claim 32 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claims 35-38

Applicants respectfully submit that claims 35-38 are dependent directly or indirectly on claim 32, thus include the same limitations as claim 32. As such, claims

35-38 are patentable for at least the same reasons as claim 32.

Claim 13

Claim 13 as amended requires a processing unit that executes a data ciphering operation. The processing unit "**prior** to the completion of the swapping of the data stored in the data structure for data ciphering of the **first portion** of the plaintext, . . . is to access data stored in the data structure for data ciphering of a **second portion** of the plaintext" Furthermore, claim 13 requires that "the processing unit is to data cipher the second portion of the plaintext upon determining that the data being swapped in the data structure does not equal the data being accessed in the data structure."

Schneier describes the inner loop of an RC4 stream cipher where swapping of S[i] and S[j] associated with an 8 bit portion of keystream must occur before a load associated with another 8 bit portion of keystream because even in a fully unrolled optimized loop of the algorithm as described by Schneier the swapping associated with an 8 bit portion of keystream must completed before a load associated with another 8 bit portion of keystream to prevent the algorithm outputting unusable ciphertext, as discussed above. Thus, Schneier fails to describe a processing unit that "**prior** to the completion of the swapping of the data stored in the data structure for data ciphering of the **first portion** of the plaintext, . . . is to access data stored in the data structure for data ciphering of a **second portion** of the plaintext."

Furthermore, Schneier fails to describe a processing unit that is to data cipher

the second portion of the plaintext upon determining that the data being swapped in the data structure does not equal the data being accessed in the data structure.

Therefore, Schneier fails to describe a processing unit to execute a data ciphering operation that "**prior** to the completion of the swapping of the data stored in the data structure for data ciphering of the **first portion** of the plaintext, . . . is to access data stored in the data structure for data ciphering of a **second portion** of the plaintext." Furthermore, Schneier fails to describe that "the processing unit is to data cipher the second portion of the plaintext upon determining that the data being swapped in the data structure does not equal the data being accessed in the data structure."

These aspects of claim 13 increase the number of process cycles that the memory is accessed over the prior art algorithm as described in Schneier. Applicant's claim 13 more fully utilizes the memory and provides for faster execution of data ciphering over that disclosed in Schneier. These aspects can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Because Schneier does not describe a processing unit that "**prior** to the completion of the swapping of the data stored in the data structure for data ciphering of the **first portion** of the plaintext, . . . is to access data stored in the data structure for data ciphering of a **second portion** of the plaintext" and "is to data cipher the second portion of the plaintext upon determining that the data being swapped in the data structure does not equal the data being accessed in the data structure," Schneier fails to anticipate claim 13.

Furthermore, Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claim 13 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claims 15-20

Applicants respectfully submit that claims 15-20 are dependent directly or indirectly on claim 13, thus include the same limitations as claim 13. As such, claims 15-20 are patentable for at least the same reasons as claim 13.

Claims 21 and 28

Applicant respectfully submits that claims 21 and 28 as amended require similar limitations as claim 13. Specifically, claim 21 as amended requires a co-processor coupled to a host processor and a host memory. The co-processor includes an interface unit, and an execution unit. The execution unit includes memory, a microcontroller, and an RC4 unit. The RC4 unit "is to swap data . . . for data ciphering of a **first portion** of plaintext and . . . read data . . . for data ciphering of a **second portion** of the plaintext, **prior** to the completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plaintext." Furthermore, claim 21 requires "the RC4 unit is to data cipher the **second portion** of the plaintext upon determining that the data being swapped in the S-box does not equal the data being read from the S-box."

Similarly, claim 28 as amended requires a system including a host processor,

a host memory, and a co-processor. The co-processor includes an interface unit, and an execution unit. The execution unit includes a memory, a microcontroller, and an RC4 unit. The RC4 unit "is to swap data stored . . . for data ciphering of a **first portion** of the plaintext and . . . is to read data stored . . . for data ciphering of a **second portion** of the plaintext, **prior** to completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plain text." Moreover, claim 28 requires "the RC4 unit is to swap data retrieved from the data structure for the data ciphering of the **second portion** of the plaintext upon determining that the data being swapped for the data ciphering of the first portion of the plaintext does not equal the data read from the data structure for data ciphering of the second portion of the plaintext."

Thus, for at least the reasons discussed for claim 13, Schneier fails to anticipate claim 21 and 28.

Furthermore, Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claims 21 and 28 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claims 23-24

Applicants respectfully submit that claims 23-24 are dependent directly or indirectly on claim 21, thus include the same limitations as claim 21. As such, claims 23-24 are patentable for at least the same reasons as claim 21.

Claims 29 and 30

Applicants respectfully submit that claims 29 and 30 are dependent directly or indirectly on claim 28, thus include the same limitations as claim 28. As such, claims 29 and 30 are patentable for at least the same reasons as claim 28.

Claim Rejections under 35 U.S.C. §103(a)

Claims 2-3, 15-16, 20, and 33-34 have been rejected under 35 USC §103(a) as being unpatentable over Schneier et al.

Claims 6, 8-12, 14, 22, 24, 29, 31, 37, and 39-44 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Schneier et al., and further in view of U.S. Patent No. 5,454,117 to Puziol et al. ("Puziol").

Claims 25-27 have been rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,873,707 to Batcher ("Batcher"), and further in view of Puziol et al.

Applicant respectfully disagrees with the rejection because the combinations fail to describe or suggest the limitations of the claims. Applicant requests reconsideration of the rejected claims.

Claim 8

Claim 8 as amended requires receiving a request to perform data ciphering of plaintext and then processing the request. The processing of the data ciphering of plaintext includes "data ciphering a **first portion** of the plaintext based on swapped

data from a first access of data." The processing further includes "performing a **second access** of data from the data structure **prior** to the swapping of the data from the **first access**." Moreover, claim 8 requires "upon determining that the data from the first access does not equal the data from the second access, . . . data ciphering a **second portion** of the plaintext based on swapped data from the second access in an iteration including data ciphering a first portion of the plaintext based on the swapped data from the first access." Furthermore, the claim requires "upon determining that the data from the first access equals data from the second access . . . **reexecuting** the performing of the second access of data from the data structure in an iteration that is subsequent to determining that the data from the first access does not equal the data from the second access."

As discussed above, Schneier describes an inner loop of an RC4 stream cipher that only loads the values from the S-box needed to calculate the value of the cipher for one bit keysteam. Furthermore, even in a fully unrolled optimized loop of the algorithm as described by Schneier the stores associated with an 8 bit portion of keystream must be completed before a load associated with another 8 bit portion of keystream to prevent the algorithm outputting unusable ciphertext.

Moreover, Schneier fails to describe data ciphering a **first portion** of the plaintext based on swapped data from a first access of data" and "data ciphering a **second portion** of the plaintext based on swapped data from the second access in an iteration including data ciphering a first portion of the plaintext based on the swapped data from the first access" because the inner loop of this prior art algorithm only loads the values from the S-box needed to calculate the value of the cipher for

one 8 bit keystream. Furthermore, Schneier fails to describe reexecuting the performing of the second access of data from the data structure in an iteration that is subsequent to determining that the data from the first access does not equal the data from the second access. In other words, Schneier fails to describe data ciphering of two portions of plaintext within one iteration and re-execution of one portion of plaintext under certain conditions.

Therefore, Schneier fails to describe or suggest data ciphering a **first portion** of the plaintext based on swapped data from a first access of data, performing a **second access** of data from the data structure **prior** to the swapping of the data from the **first access**, upon determining that the data from the first access does not equal the data from the second access, data ciphering a **second portion** of the plaintext based on swapped data from the second access in an iteration including data ciphering a first portion of the plaintext based on the swapped data from the first access, and upon determining that the data from the first access equals data from the second access . . . **reexecuting** the performing of the second access of data from the data structure in an iteration that is subsequent to determining that the data from the first access does not equal the data from the second access, as required in claim 8.

These aspects of claim 8 increases the number of process cycles that the memory is accessed over the prior art algorithm as described in Schneier. Applicant's claim 8 more fully utilizes the memory and provides for faster execution of data ciphering over that described in Schneier. These aspects can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an

example of an embodiment and not provided as a limitation thereof.

Puziol describes configurable branch prediction hardware for a processor, comprising logic and interconnect, which is configurable via a control line. The branch prediction logic is made up of branch prediction cache, next state logic, predicted direction logic, write address generator, history ram, and read address generator. This hardware is used to predict the target address of the next set of target instructions for the processor to prevent pipeline bubbles in a processor.

The hardware makes a static and dynamic prediction for each branch. The static prediction is based on the branch's opcode. Unconditional branches have a static prediction of taken. Conditional branches have static prediction of not-taken. Prediction direction logic is used to resolve if a conditional loop instruction is given a prediction of taken or not-taken. The dynamic predictions are made at run time. In the case the branch prediction logic mispredicts the target address, the "correct" instruction bytes are read from the branch prediction cache and the mispredicted address is aborted.

Puziol fails to describe or suggest data ciphering a **first portion** of the plaintext based on swapped data from a first access of data, performing a **second access** of data from the data structure **prior** to the swapping of the data from the **first access**, upon determining that the data from the first access does not equal the data from the second access, data ciphering a **second portion** of the plaintext based on swapped data from the second access in an iteration including data ciphering a first portion of the plaintext based on the swapped data from the first access, and upon determining that the data from the first access equals data from

the second access . . . **reexecuting** the performing of the second access of data from the data structure in an iteration that is subsequent to determining that the data from the first access does not equal the data from the second access, as required in claim 8.

Furthermore, The branch prediction hardware of Puziol may predict an instruction to perform a second access will be requested by the algorithm running on the processor. However, the algorithm must actually cause the processor to perform a second access; otherwise, a second data access will never be preformed. Similarly, the branch prediction hardware may predict instructions to perform data ciphering a second portion of the plaintext or to re-execute one portion of plaintext will be required by the algorithm running on the processor, but without the algorithm of Schneier actually causing the processor to perform data ciphering of two portions of plaintext within one iteration and to re-execute one portion of plaintext under certain conditions the branch prediction hardware cannot cause these instructions to be preformed. Since Schneier fails to describe or suggest the above, the combination of Puziol and Schneier cannot describe or suggest the requirements of claim 8.

Because Schneier and Puziol do not describe or suggest data ciphering a **first portion** of the plaintext based on swapped data from a first access of data, performing a **second access** of data from the data structure **prior** to the swapping of the data from the **first access**, upon determining that the data from the first access does not equal the data from the second access, data ciphering a **second portion** of the plaintext based on swapped data from the second access in an

iteration including data ciphering a first portion of the plaintext based on the swapped data from the first access, and upon determining that the data from the first access equals data from the second access . . . **reexecuting** the performing of the second access of data from the data structure in an iteration that is subsequent to determining that the data from the first access does not equal the data from the second access, the combination of Schneier and Puziol fails to render claim 8 obvious.

Claims 11 and 12

Applicants respectfully submit that claims 11 and 12 are dependent directly or indirectly on claim 8, thus include the same limitations as claim 8. As such, claims 11 and 12 are patentable for at least the same reasons as claim 8.

Claim 39

Applicant respectfully submits that claim 39 as amended requires similar limitations as claim 8. Specifically, claim 39 as amended requires processing that comprises "data ciphering a **first portion** of the plaintext based on the swapped data from the first access." The processing further comprises "performing a **second access** of data from the data structure **prior** to the swapping of the data from the **first access**." Moreover, claim 39 requires "upon determining that the data from the first access does not equal the data from the second access, . . . data ciphering a second portion of the plaintext based on the swapped data from the second access in an iteration including data ciphering a first portion of the plaintext based on the

swapped data from the first access." Furthermore, the claim requires "upon determining that the data from the first access equals data from the second access . . . **reexecuting** the performing of the second access of data from the data structure in an iteration that is subsequent to determining that the data from the first access does not equal the data from the second access."

Thus, for at least the reasons discussed for claim 8, the combination of Schneier and Puziol fails to render claim 39 obvious.

Claims 43 and 44

Applicants respectfully submit that claims 43 and 44 are dependent directly or indirectly on claim 39, thus include the same limitations as claim 39. As such, claims 43 and 44 are patentable for at least the same reasons as claim 39.

Claim 25

Claim 25 requires an apparatus including a memory and an RC4 hardware state machine. The two are coupled together "to generate a plurality of output text blocks from a plurality of input text blocks." The RC4 state machine generates "a subset of said plurality of output text blocks . . . as a result of repeating the same sequence of states." During these "sequence of states data is **speculatively** read . . . as part of the generation of a **next one** of said plurality of output text blocks **prior** to a write . . . completing as part of generation of a **current one** of said plurality of output text blocks."

Batcher describes a cycle stealing hardware configuration that contributes to

faster processing of the S-box table initialization phase and the decrypt/encrypt phase of an RC4 algorithm. Batcher accelerates operations of a microcode controller by using a level sensitive address latch and a level sensitive instruction code world latch to create the effect of a dual ported microcode storage, which allows the fetch of the next microcode word and current microcode execute operation to proceed on the same clock cycle. This configuration allows the clock frequency of the system to be increased by twice the frequency as the prior art because of the setup/hold relationship to the latches are improved. (Batcher, col. 10, ll. 12 – 15).

That is this cycle stealing hardware configuration uses the leading and falling edge of a clock signal to speed up the data accessing. However, the hardware configuration does not "**speculatively** read . . . as part of the generation of a **next one** of said plurality of output text blocks **prior** to a write . . . completing as part of generation of a **current one** of said plurality of output text blocks." (claim 25). Furthermore, cycle stealing hardware configurations, such as Batcher, do not move loads in front of stores to optimize RC4 algorithms.

As discussed above, Puziol describes configurable branch prediction hardware for a processor used to predict the target address of the next set of target instructions for the processor to prevent pipeline bubbles in a processor. Puziol fails to describe an RC4 state machine that generates "a subset of said plurality of output text blocks . . . as a result of repeating the same sequence of states" in which "data is **speculatively** read . . . as part of the generation of a **next one** of said plurality of output text blocks **prior** to a write . . . completing as part of generation of a **current**

one of said plurality of output text blocks." (claim 25).

Furthermore, branch predictors, such as Puziol, determine the proper address for the next set of instructions prior to the processor making this request to optimize the efficiency of a processor. Branch predictors do not move loads in front of stores to optimize RC4 algorithms.

Because Batcher and Puziol do not describe an RC4 state machine that generates "a subset of said plurality of output text blocks . . . as a result of repeating the same sequence of states" in which "data is speculatively read . . . as part of the generation of a next one of said plurality of output text blocks prior to a write . . . completing as part of generation of a current one of said plurality of output text blocks," the combination of Batcher and Puziol fails to render claim 25 obvious.

Claim 26 and 27

Applicant respectfully submits that claims 26 and 27 depend on independent claim 25 and include all the limitations of claim 25. As such, the combination of Batcher and Puziol fails to render claims 26 and 27 obvious for at least the same reasons as claim 25.

Conclusion

Applicant respectfully submits that the rejections have been overcome by the remarks. Accordingly, Applicant respectfully requests the rejections be withdrawn and the claims allowed. If the allowance of these claims could be facilitated by a telephone conference, the Examiner is invited to contact the undersigned at (408) 720-8300. If there are any additional charges, please charge our Deposit Account No. 02–2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: 6/19, 2006

Daniel M. De Vos
Registration No. 37,813

Customer No. 08791
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1030
(408) 720-8300